Copy 19 of 33 copies

2

AD-A205 302

IDA MEMORANDUM REPORT M-553

SUPPORT TO THE PHASE ONE ENGINEERING TEAM 1988

Cathy J. Linn
Cy D. Ardoin
Dennis W. Fife
Karen D. Gordon

January 1989

Prepared for
Strategic Defense Initiative Organization

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

IDA Log No. HQ 88-34024

The work reported in this document was conducted under contract MDA 903 84 C 0031 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release, unlimited distribution.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) IDA Memorandum Report M-553	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Institute for Defense Analyses	6b OFFICE SYMBOL IDA
7a NAME OF MONITORING ORGANIZATION OUSDA, DIMO	
6c ADDRESS (City, State, and Zip Code) 1801 N. Beauregard St. Alexandria, VA 22311	7b ADDRESS (City, State, and Zip Code) 1801 N. Beauregard St. Alexandria, VA 22311
8a NAME OF FUNDING/SPONSORING ORGANIZATION SDIO	8b OFFICE SYMBOL (if applicable) SDIO
9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA 903 84 C 0031	
8c ADDRESS (City, State, and Zip Code) Room 1E149, The Pentagon Washington, D.C. 20301-7100	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
		T-R2-578	

11 TITLE (Include Security Classification)
Support to the Phase One Engineering Team 1988 (U)

12 PERSONAL AUTHOR(S)
Cathy J. Linn, Cy D. Ardoin, Dennis W. Fife, Karen D. Gordon

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM ______ TO ______	1989 January	50

16 SUPPLEMENTARY NOTATION

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Software tools and techniques; computer architecture; modeling and simulation; data modeling techniques; distributed operating systems.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of this IDA Memorandum Report is to document information, recommendations and guidance delivered to the Phase One Engineering Team (POET) during the time period from January 1988 to December 1988. This document consists of briefings, white papers and memoranda, collected to document support to the POET during the year.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include area code)	22c OFFICE SYMBOL
Dr. Cathy Jo Linn	(703) 824-5520	IDA/CSED

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

IDA MEMORANDUM REPORT M-553

SUPPORT TO THE PHASE ONE ENGINEERING TEAM 1988

Cathy J. Linn
Cy D. Ardoin
Dennis W. Fife
Karen D. Gordon

December 1988

INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 84 C 0031
Task T-R2-578

Preface

The enclosed documents represent information, recommendations, and guidance delivered to the Phase One Engineering Team (POET) during the time period from January 1988 to December 1988. These are provided in the form of briefings, white papers, and memorandum. They are collected in one IDA Memorandum to document our support to the POET during the year.

To: J. Dominitz

From: D. W. Fife, IDA-CSED

Date: 16 February 1988

Subject: Alternatives for Tools and Architecture Representation

As you requested, I have composed a chart comparing alternative tools according to the requirements we discussed.

I have eliminated TAGS and DCDS by restricting consideration to the Sun workstation platform, which SDIO currently uses and which current BM/C3 contractors have used in Phase IIC. With the PC-AT platform currently used for DCDS graphics, it can also be eliminated for inadequate quality. The candidates I propose for discussion are AUTO-G, Teamwork, and Software Through Pictures. I have considered only currently delivered tool capabilities.

IDA's current tracking of computer-aided tools convinces us that a highly competitive market is producing major tool enhancements and, consequently, comparative cost-benefit realignments among tools. POET would NOT be well advised to recommend a specific tool other than for its direct use in the near future. A better approach is to define minimum tool requirements to guide a tool selection by the contractor or staff that will directly use the tool for POET's support. These requirements also would help tool producers set their enhancement goals so as to assist SDI programs better in the future. The attached table comparing the three candidates indicates such requirements by a short phrase in the left hand column. My previous memo can be used for a fuller explanation of each requirement.

THREE ALTERNATIVES FOR POET ARCHITECTURE REPRESENTATION TOOLS AND HOW THEY MEET THE REQUIREMENTS

REQUIREMENT OR FUNCTION	TOOL ALTERNATIVE		
	AUTO-G	Teamwork	Software Through Pictures
hierarchical decomposition	hierarchy via branches on one tree diagram	hierarchy of separate diagrams	hierarchy of separate diagrams
asynchronous processes	unique icon on each process branch of tree	ambiguous unless control flow shown	ambiguous unless control flow shown
data flow depiction	distinct input & output icons on branches	diagrams depict flow lines between processes	diagrams depict flow lines between processes
timing	distinct icons on process branches	not available	not available
sequencing of processes	control messages and process state responses	separate control and state diagrams	separate control and state diagrams
critical function sequences	user-written software	user-written software	user-written software
physical resources *and allocations*	fixed by designer *in separate processes,* aided by replication	fixed by designer *in separate processes,* no replication aid	fixed by designer *in separate processes,* no replication aid
sizing and other performance analysis	user written code or simulation	user written code or simulation	user written code or simulation
communications protocols & routes	broadcast & point to point routes; asynchronous protocols built-in	broadcast & point to point routes impractical if extensive; no protocols	broadcast & point to point routes impractical if extensive; no protocols
typed data items	8 predefined types	user defined type names	user defined type names
data structures	arrays, records, and user defined complex types and templates	user defined type names	user defined type names
data dictionary search	query diagram or view separate list	selective query on separate menu	selective query on separate menu
data structure diagram	tree form on one diagram	separate ERA or tree form	separate ERA or tree form
arithmetic processing	flow chart on same diagram	separate user written code	separate user written code
produce executable simulation code	SADMT language text generated from diagram automatically	user enters code as text, but tool doesn't process it	user enters code as text, but tool doesn't process it

THREE ALTERNATIVES TOOLS
AND HOW THEY MEET
THE REQUIREMENTS
(continued)

REQUIREMENT OR FUNCTION	TOOL ALTERNATIVE		
	AUTO-G	Teamwork	Software Through Pictures
diagnostic checking	complete to flow chart level, with easy error location	consistency of flows and names on each diagram type	consistency of flows and names on each diagram type
print diagrams lists and tables	screen dump or plot in pieces	screen dump or Postscript printer	screen dump or Postscript printer
prepare standard documents	no direct support	DoD specs from Document Production Interface	no direct support yet
track design versions	automatic and user labeled	automatic up to 16 versions	use separate packages
manage design configurations	user administered	user administered	user administered
protect design parts	administer by OS protection	any user locking	authorized user locking
user-added graphic icons	none	yes, by graphic notes	none
design database access	yes, but support uncertain	yes, and access module protects database damage	yes, user responsible for protection
user tailored edit templates	not applicable	yes	yes
easy to use and few graphic annoyances	okay, but scrolling is tedious	okay	okay, but invisible items cause trouble
user manual quality	good, with practical examples	best of the three	okay, but needs an index
experience base on SDI	Martin Marietta on BM-C3 and vendor on sensor data fusion	SAIC has used for BM-C3	TRW and Sparta used on BM-C3
cost	approx $25K for first copy	approx $14K for first copy	approx $17K for first copy

POET
ARCHITECTURE COMPOSITION TASK

Report on A-SPEC Support Tasking

Dennis Fife, IDA, Presenter
Frank Frangione, Mitre, Chairman
Kathie Groveston, Mitre

A-SPEC SUPPORT RECOMMENDATIONS

- A-SPEC graphic requirements
- Comparison of SDL and SRA to A-SPEC graphic needs
- Comparison of Software Thru Pictures to A-SPEC needs and SDL/SRA
- STP augmentation
- Implementation Factors

A-SPEC GRAPHIC INFORMATION REQUIREMENTS
Geospatial system layout identifying platform types and numbers, relative locations, and orbital characteristics.
Interplatform information flow diagram, depicting platform types and aggregate information flows among them.
SDS operating modes or states and transition criteria.
Intersystem (SDS, Blue offense, etc.) information flow diagram.
Descriptions of inputs, functions, and outputs at successive levels of detail and associated with physical elements.
Object descriptions to include system performance *requirements and allocated* performance levels.
Presentations of object component interrelationships (e. g. message types, management structures, element components)
Description of data and message flows related to physical communications topology and interconnections.
End-to-end functional control sequencing.
End-to-end functional time budgeting.
General line drawings as appropriate for explanatory purposes.

A-SPEC REQUIREMENTS COMPARED TO SDL AND SRA

A-SPEC GRAPHIC INFORMATION REQUIREMENT	SPECIFIED PRESENTATION FORM	
	SDL	SRA
Geospatial system layout identifying platform types and numbers, relative locations, and orbital characteristics.	None	None
Interplatform information flow diagram, depicting platform types and aggregate information flows among them.	SDL IDEF-0 diagram with platform viewpoint	Schematic block diagram
SDS operating modes or states and transition criteria.	SDL state transition diagram	RAS (not graphic)
Intersystem (SDS, Blue offense, etc.) information flow diagram.	SDL IDEF-0 F-1 diagram	Schematic block diagram
Descriptions of inputs, functions, and outputs at successive levels of detail and associated with physical elements.	SDL IDEF-0 decomposition including mechanisms	RAS (not graphic)
Object descriptions to include system performance requirements and allocated performance levels.	SDL data dictionary	RAS (not graphic)
Presentations of object component interrelationships (e. g. message types, management structures, element components)	SDL entity relationship diagrams or structure charts	RAS (not graphic)
Description of data and message flows related to physical communications topology and interconnections.	None	RAS (not graphic)
End-to-end functional control sequencing.	SDL IDEF-2	FFBD
End-to-end functional time budgeting.	SDL IDEF-2	Timeline diagram
General line drawings as appropriate for explanatory purposes.	None	None

COMPARISON OF STP SUPPORT TO A-SPEC NEEDS AND SDL

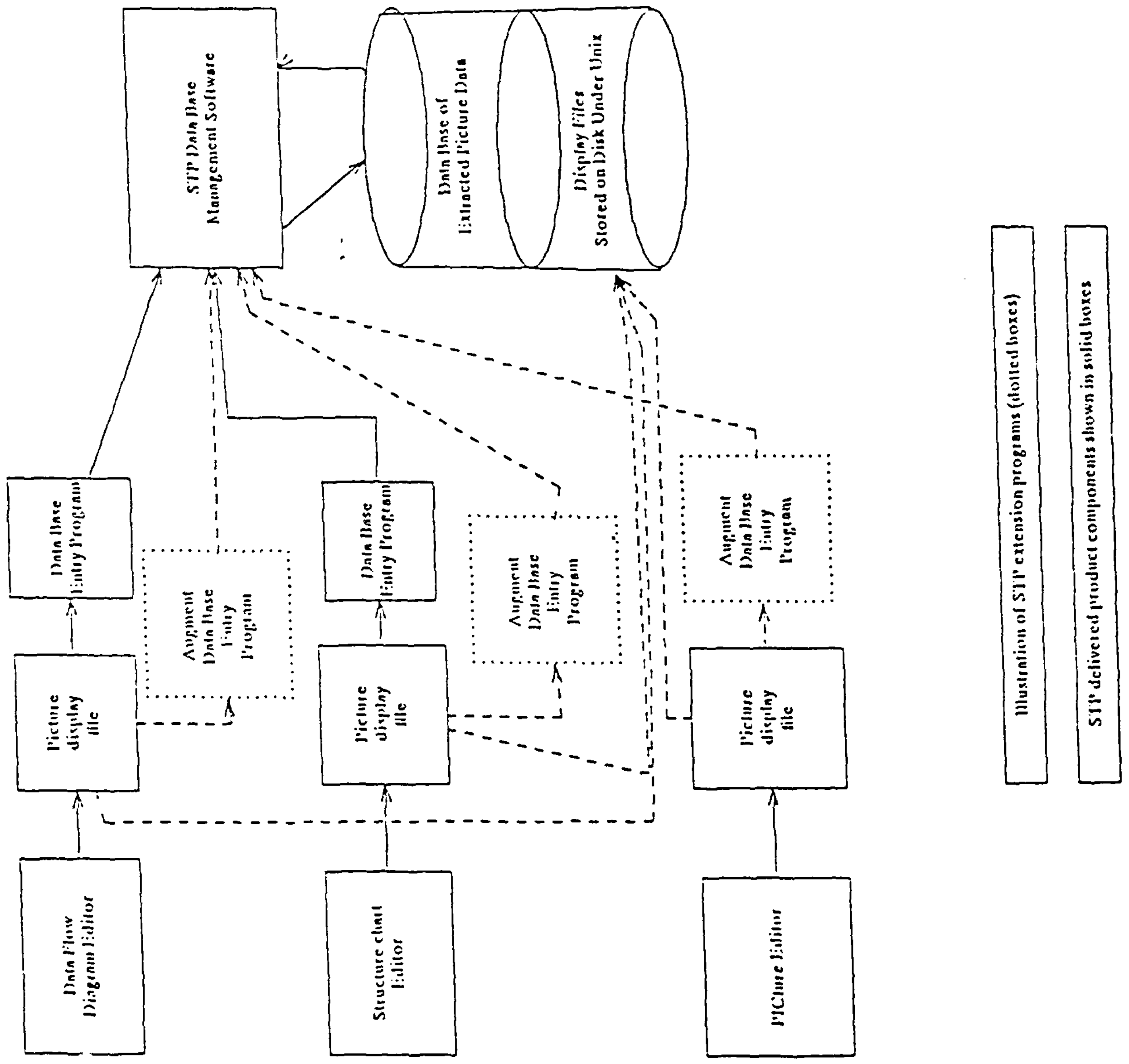
A-SPEC REQUIREMENT	SDL FORM	STP SUPPORT FOR A-SPEC NEED/SDL FORM
Geospatial system layout identifying platform types and numbers, relative locations, and orbital characteristics.	None	Inadequate. PIC editor supports crude schematics with a limited number of shapes. Labels can be placed anywhere. Information is NOT put into STP database.
Interplatform information flow diagram, depicting platform types and aggregate flows.	SDL IDEF-0 diagram with platform viewpoint	YES, using the Data Flow Diagram Editor
SDS operating modes or states and transition criteria.	SDL state transition diagram	YES, using the State Transition Diagram Editor
Intersystem (SDS, Blue offense, etc.) information flow.	IDEF-0 F-1 diagram	YES, using the Data Flow Diagram Editor
Descriptions of inputs, functions, and outputs at successive levels of detail and associated with physical elements.	IDEF-0 decomposition including mechanisms	YES, using the Data Flow Diagram Editor and indicating mechanisms by *position*
Object descriptions to include system performance requirements and allocated performance levels.	SDL data dictionary	YES, supplemented with diagram comments
Graphical presentations of object component interrelationships (e. g. message types, management structures, element components)	SDL entity relationship diagrams or structure charts	YES, with the ERA Diagram Editor or Data Structure Editor
Graphical description of data and message flows related to physical communications topology and interconnections.	None	YES, using PIC Editor
End-to-end functional control sequencing.	SDL IDEF-2	Feasible -- special use of Data Flow Diagram Editor, not IDEF-2
End-to-end functional time budgeting.	SDL IDEF-2	Feasible – special use of Data Flow Diagram Editor with diagram comments, not IDEF-2
General line drawings as appropriate for explanatory purposes.	None	Inadequate using the PIC Editor

COMPARISON OF STP SUPPORT TO A SPEC NEEDS AND SRA

A-SPEC GRAPHIC INFORMATION REQUIREMENT	SRA FORM	STP SUPPORT FOR A-SPEC NEED/SRA FORM
Geospatial system layout identifying platform types and numbers, relative locations, and orbital characteristics.	None	Inadequate, using PIC Editor.
Interplatform information flow diagram, depicting platform types and aggregate information flows among them.	Schematic block diagram	YES, *using DFD Editor.*
SDS operating modes or states and transition criteria (graphic preferred).	RAS (not graphic)	YES, using the State Transition Diagram Editor; or, NO, for RAS – use word processing.
Intersystem (SDS, Blue offense, etc.) information flow diagram.	Schematic block diagram	YES, using DFD Editor for SDL.
Descriptions of inputs, functions, and outputs at successive levels of detail and associated with physical elements.	RAS (not graphic)	YES, using DFD Editor for SDL; or, NO for RAS – *use word processing.*
Object descriptions to include system performance requirements and allocated performance levels.	RAS (not graphic)	YES, using data dictionary and diagram comments; or, NO, for RAS – use word processing.
Presentations of object component interrelationships (e. g. message types, management structures, element components)	RAS (not graphic)	YES, using SDL structure charts; or, NO for RAS – use word processing.
Description of data and message flows related to physical communications topology and *interconnections.*	RAS (not graphic)	YES, using PIC Editor; or, NO for RAS – use word processing.
End-to-end functional control sequencing.	FFBD	Feasible with special use of DFD Editor; or NO, using FFBD.
End-to-end *functional* time budgeting.	Timeline diagram	Feasible with special use of DFD Editor; or, NO for SRA timeline diagram.
General line drawings as appropriate for explanatory purposes.	None	Inadequate, using PIC Editor

STP AUGMENTATION

- Use existing editors without modification (per POET guideline)
  - Data Flow Editor
  - Structure Chart Editor
  - PIC Picture Editor
- Extend underlying database structure to:
  - capture all specifications
  - support validation queries and improved consistency checking
- Develop additional programs to:
  - capture all data from diagrams
  - provide added consistency checks


STP Data Base Management Software
Data Base of Extracted Picture Data
Display Files Stored on Disk Under Unix
Data Base Entry Program
Augment Data Base Entry Program
Data Base Entry Program
Augment Data Base Entry Program
Augment Data Base Entry Program
Picture display file
Picture display file
Picture display file
Data Flow Diagram Editor
Structure chart Editor
PICture Editor
Illustration of STP extension programs (dotted boxes)
STP delivered product components shown in solid boxes

STP IMPLEMENTATION CONSIDERATIONS

- Phase IIC Assets Available to Redeploy
  - Recommended for SETA Contractor and POET (each)
    - 1 Sun server
    - 2 Workstations
    - 1 Laser printer
- Guideline being drafted by POET

GUIDELINE ELEMENTS

- Use SDL where its representations are relevant to A-SPEC needs
- Use STP, with augmentation, to implement SDL
- Key portions of A-SPEC will be produced with STP+
- SRA process will use SDL and STP+

To: POET Architecture Composition Group

From: D. W. Fife

Date: Sept. 9, 1988

Subject: Recommended interactions with GE

This memo documents the technical steps of POET and GE interaction, as I defined them in our July 21 meeting, and which GE accepted and has begun to implement.

GE primarily will use the RDD tool, rather than the SDL description language, because of the more extensive specification capabilities of RDD language. They have agreed to provide output presentations in SDL form to POET, and to accept that form as input to their integrated database. The proposed interactions are intended to ensure that effective and reliable interchange occurs, on an automated basis insofar as practical.

1. GE will also use Software Through Pictures, or DesignIDEF, as POET does.
2. GE will edit POET submissions as necessary to achieve input to RDD.
3. GE will create "equivalent" RDD specifications and diagnose their correctness with RDD.
4. GE will automatically produce IDEF0 diagrams from the RDD database.
5. GE will provide POET with:
   a. the edited versions of POET IDEF0 diagrams used for RDD input,
   b. the RDD diagnostic analysis report,
   c. the automated IDEF output from RDD, which may differ from a.

GE has begun to test and implement this concept with the Sparta architecture from Phase IIC, as evidenced by their results at the August 31 meeting.

To: J. Dominitz, POET

From: D. W. Fife [illegible]

Date: Sept. 7, 1988

Subject: Documents Review

These are my observations/conclusions from the three document packages (SRS, Sparta, and TASC/ATI) that Kathy sent to me.

SRS Final Report Briefing. SRS has made a useful contribution in assessing the four contractor results, and transforming them into one composite architecture for the four cited aspects of BM/C3. However, I have not seen the full report, but only the tracking area as presented in these briefing charts. I do not find an accompanying rationale that would justify adopting the SRS composite architecture as POET's starting point rather than Sparta's.

The four SRS functions at the top level (A0) are very different from Sparta's. This illustrates the problem of IDEF perspective. SRS criticizes the Phase IIC results on their focus and understandability, asserting that this is directly due to the way the method was used. But, there is no known way to guarantee that any IDEF analysis (or any other structured analysis method) doesn't have the same faults relative to some reviewer's viewpoint. Moreover, there is no way to clearly and easily compare two IDEF0 results, say Sparta and SRS, except by producing new consensus diagrams from a box-by-box and flow-by-flow review and revision.

I do not agree with the conclusions that SRS states. It isn't apparent to me that IAA per se was key to understanding what the Phase IIC contractors accomplished or enabling the extraction of their best work. Sure enough, SRS manpower achieved these goals, but it doesn't follow that another team, given the contractor results and the IAA method, would come up with the same or similar composite architecture.

SRS evaluated contractor results relative to one another and to its own perception of IDEF quality. This does not imply any absolute measure of quality of the designs per se. Using the term "comprehensive" for an evaluation level makes it seem that the results were sufficient for an SDS design. I would disagree if SRS meant that. I have reviewed Sparta's results fairly well, and taking it as representative of all Phase IIC results, I feel that SDIO needs much more thorough and precise formulations. SRS doesn't state succinctly why their composite is so different from the contractors' architectures, or why it is a better expression of an SDS design. Though it may be better IDEF0, and is clearer to me personally at the tracking level, it does not obviously portray a more understandable, complete, or effective SDS architecture than the other contractors.

The SRS report points up once more the need for POET to press on with the intensive review and interchange that will produce a widely agreed (and therefore, validated) perspective of SDS architecture expressed in IDEF0. To be most effective, the resulting IDEF0 description needs a lot more precision and depth than shown in the examples to date.

The briefing doesn't identify specific directions for improving SDL; see Sparta comments below.

Sparta. This set of briefing charts documents a very short study done of a long list of complex issues, so its results on many are fairly shallow. In brief, there are two parts: one done by a former SRS employee as a consultant on SDL; the second by TBE on using TAGS or DCDS as a bridge from SDL to simulation. The latter is not properly focused. It gives no evidence that the present simulation facilities of TAGS and DCDS are adequate for SDI needs, and bridging per se is not an issue. IDA has already demonstrated techniques for bridging from IDEF0 to SADMT simulation on Software Through Pictures (forthcoming IDA paper by David Wheeler). The primary need in creating any such bridge is to provide the additional specifications that SDL doesn't express, which is why GE has chosen to use DCDS and its commercial cousin, RDD. This brings us to the first part of this report – SDL limitations.

The report stresses the need for better use of IDEF0 mechanism, and I agree. Mechanism MUST be used to identify the physical components and processing resources that implement the logical functions shown. Sparta also argues for a platform orientation in initial decomposition. This also is persuasive, because it fits the current mind-set and crucial issues of the SDI program, and as an added benefit, suits the SADMT simulation framework. Sparta provided such a diagram, and there is no reason why it cannot serve as the top level IDEF0 diagram.

Other possible extensions mentioned for SDL are not defined well enough to warrant funding. The call for more "research" is particularly vague and self-serving. In my view, briefly, the most important needs are:

- tools that integrate all specification elements in one database structure, as we recommended to Mr. Israel in June;
- formal, rigorous means to specify timing and control, improving on extensions already widely used in Yourdon-DeMarco and similar methods;
- formal coupling of IDEF0 mechanism to other rigorous block or schematic diagrams that can depict complex physical and dynamic relationships, such as parts explosion, geospatial topology, etc.

TASC/ATI Decomposition. The voluminous TASC report provides a new tree of SDS functions, and illustrates the use of FFBDs and RAS. It doesn't yet have many of the IDEF0 diagrams promised to accompany the other material. The functions chosen are arguable, and again, no rationale is presented as to why they are better choices than, say, Sparta's. The FFBDs, it becomes clear, depict the same functions as one would put in an IDEF0 diagram, but show which of them may execute concurrently. But, the diagrams do not show the input and output data, nor the events or data that trigger the performance of the functions. The SRA advocates vigorously argued its advantages in traceability of requirements, so I was astounded that this report has absolutely NO information for that purpose. This report may be useful to POET and GE in coming to ONE decomposition for adoption throughout SDI, but the report does not change my view that SRA is less informative, more poorly organized, and otherwise redundant with SDL.

March 8, 1988

A Simulation Framework for the Strategic Defense System

Cathy Jo Linn

Computer and Software Engineering Division
Institute for Defense Analyses
Alexandria, Virginia

SUMMARY

This report briefly discusses the requirements of a simulation framework for the Strategic Defense System (SDS) and the current efforts that address these requirements. A comparison of these efforts is given and used to motivate a plan for the establishment and evolution of a Simulation Framework for the SDS.

The conclusion is that DETEC should be the initial operating capability of the SDS Simulation Framework. However, the long-term requirements of the SDS Simulation Framework must be met by drawing on technology developed by the other simulation efforts underway. Specifically, the evolution must provide:

(1) a higher level of abstraction for simulation operations,

(2) a model of communication that supports distribution,

(3) a distributed implementation of the simulation driver,

(4) a link to the design process, and

(5) an Ada-based implementation of the interface specification.

1. INTRODUCTION

The purpose of this document is to propose a detailed technical plan for the establishment and evolution of a standard SDS Simulation Framework. The need exists for a simulation framework that provides:

(1) simulation of a formal, unambiguous description of SDS architectures and threats,

(2) distributed simulation at all levels of fidelity, including mixed levels and the replacement of software models by actual hardware,

(3) flexibility for the simulation of all existing SDS architectures as well as potential future designs,

(4) a set of interfaces to the environment and approved technology models that allows easy identification of all modeling assumptions,

(5) a substantial set of SDIO (Strategic Defense Initiative Office) approved and documented technology models for all elements currently under consideration for inclusion in SDS architectures,

(6) a library of well documented battle management, command and control algorithms available for reuse in architectures with compatible functional interfaces, and

(7) a user interface to support the construction and modification of architectural models.

Such a simulation framework *will not* eliminate the design problem of the SDS. It *will not* standardize a set of functional interfaces. It *will* provide a set of standard simulation interfaces and enable the evaluation of alternative SDS architectures.

Several organizations are studying the issues involved in such a simulation framework. These efforts have originated and developed with somewhat divergent goals, but appear to have arrived at similar conclusions. A brief description and comparison of these efforts will be given followed by a proposed scheme for cooperation and interaction to maximize the use of available technical expertise in the development and evolution of the SDS Simulation Framework.

2. CURRENT EFFORTS

The following efforts are those that have been identified as addressing the need of a standard simulation framework for the SDS and the distributed implementation of such a framework. Additional efforts exist that address fundamental distributed simulation technology, the simulation of specific SDS architectures, user interfaces to a simulation, and other simulation issues. While these efforts are related, and may yield results that can be applied in the SDS Simulation Framework, direct cooperation and interaction with these efforts is not required at this time.

2.1. National Test Bed/DETEC

The National Test Bed (NTB) is a distributed facility for the test, evaluation, and maintenance of the SDS. Plans indicate that it will include a wide variety of capabilities that include distributed, end-to-end simulations, high fidelity simulation of system elements, non-functional evaluation tools, configuration management tools, and system maintenance tools.

The NTB has identified the need for a standard simulation framework and has funded work at Los Alamos National Laboratory (LANL). The Defensive Technology Evaluation Code (DETEC) group has developed a flexible simulation framework with a structure to facilitate the identification of environmental and technological assumptions made in the modeling of an architecture. Models of technology and battle management algorithms can be developed and formally specified for simulation at any level of fidelity.

The simulation driver for the framework with an extensive user interface is currently written in Fortran 77 and executes on a Cray XMP computer. Fortran 77 is also used as the formal specification language for the architecture to be simulated. An initial set of reusable technology models and battle management algorithms is being developed and expected to be available in early 1988.

2.2. The Experimental Versions

The Army is developing an experimental version of a ground-based SDS expected to be available in 1988 (EV-88). A distributed simulation testbed is being developed to support the execution and evaluation of EV-88.

A similar effort (EV -1) is being initiated by the Air Force for a space-based SDS. While this effort is in the very early stages, it is clear that a simulation testbed will be needed.

2.3. SADMT

The Battle Management Directorate of the SDIO recognized the need for (1) a formal representation of the design of SDS architectures and (2) a link between this design representation and a simulation framework for evaluation. The Strategic Defense Initiative (SDI) Architecture Dataflow Modeling Technique (SADMT) was developed by the Institute for Defense Analyses to address these requirements. This technique provides an abstract dataflow process model for the specification of battle management algorithms and technology models. These algorithms and models can be specified at any level of fidelity for simulation. As in DETEC, its structure facilitates the identification of environmental and technological assumptions made in the modeling of an architecture.

The initial prototype of the SADMT Simulation Framework is implemented in Ada on a Sun WorkStation. Ada is also used as the formal specification language for the architecture to be simulated. Work is planned to extend the prototype, distribute the driver, and provide a user interface.

SADMT was used by the Phase II-C contractors for the formal specification of their architectures. Recent deliverables from this phase are expected to provide input to the evolution of the modeling technique and the SADMT Simulation Framework. Deliverables may also provide an initial library of reusable battle management algorithms and technology models.

2.4. EVP

EVP (Experimental Version Prototype) of MITRE is an Ada-based simulation framework implemented on a shared memory multiprocessor. The emphasis of this effort was on the use of Ada and the distribution of the framework. Only limited information on EVP was available for incorporation into this draft.

2.5. EXEC

Martin Marietta's EXEC also emphasizes the support of distributed simulation. It was developed for simulations of a specific class of SDS architectures but provides a general message passing facility that could be used to support the distribution of other software packages, including a simulation framework.

3. COMPARISON OF CURRENT EFFORTS

The efforts described above each address the problem of a *simulation framework from* a different perspective and in a different timeframe. Each effort addresses its goal in a unique, yet appropriate way. A summary of the strengths and weaknesses of each effort is given in Table 1. Key elements of these differences are discussed below.

3.1. Pluggable Framework

The ability to accept an unconstrained variety of appropriately represented models is key to meeting the SDS evaluation requirements. As the SDS system continues to evolve, new technologies and new management approaches will evolve that require the incorporation of new models and algorithms into the existing simulation framework.

The DETEC and SADMT efforts pursued extensive research to determine the appropriate structure of a simulation framework and its interfaces. Although each of these efforts was performed without knowledge of the other, the structure of both systems and the abstraction of the interfaces provided is virtually identical. The major difference of abstraction lies in the modeling of communication.

EV-88 provides a degree of "pluggability" but often requires additional programming to support a new interface. This approach was appropriate for the EV-88 effort because of the requirement for a short-term deliverable. The development of the simulation driver could not be delayed for research into the "best" and most flexible interfaces for a simulation framework in the same way that the experimental version could not be delayed until the "best" architecture was determined.

3.2. Availability of Models

It is important that a wide variety of pre-programmed and verified models be available for use in the SDS Simulation Framework. The DETEC system currently has the largest library of models and algorithms. However, substantially more must be developed. It is important that as

	DETEC	SADMT	EXEC	EVP	EV 88
Pluggable Framework	yes	yes	no	yes	limited
Availability of models	yes	no	no	no	yes
Links to Design Process	no	yes	no	no	no
Supports Distribution	yes	yes	yes	yes	yes
Existing Distributed Implementation	no	no	yes	yes	yes
Explicit BMC^3	yes	extended	no	yes	limited
Base Programming Language	Fortran-77	Ada	C	Ada	Fortran,C

Table 1. Comparison of Simulation Efforts

new models are developed they can be used not only in DETEC but also in the evolving SDS Simulation Framework. Appropriate guidelines for the development of these models can provide techniques to ensure minimal changes are required.

3.3. Link to Design Process

The SADMT effort is the only simulation framework that addresses the issue of linking the design process to the simulation and evaluation process. Currently simulations are handcoded from written documents describing an architectural design. Additional assumptions about the architecture are often provided by the simulation designer. The process of translating a design into a running simulation often takes months or even years and no traceability to the original design is available. Only the simulation writers know the exact assumptions on which the simulation is based.

The SADMT notation, however, is currently supported (generated by) a commercially available design tool (Auto-G of ASA Ltd.) and other design tool vendors are currently implementing SADMT generators. This link will increase both productivity and traceability by generating the structure of the simulation directly from the design specification.

3.4. Execution Speed/Distribution

The DETEC effort was driven by the need to support large-scale simulations on the most powerful computers available. Their use of Fortran 77 on a Cray currently provides the fastest execution speed of a simulation framework.

However, to support the real-time performance that is required for SDS, the Simulation Framework must evolve to a distributed execution environment. The issue of distribution was addressed directly by EV88 and EXEC. Both provide a layer of support between the implementation of a process model for a simulation framework and the execution environment as shown in

Figure 1. Such a layered approach would allow the structure of the SDS Simulation Framework to remain constant as the implementation moved from a uniprocessor to a distributed environment.

The message-based approach of the SADMT and DETEC frameworks supports the insertion of a distributed simulation driver. Communication between processes is provided with *emit* and *consume* type operations. Currently the uniprocessor drivers of each of these systems routes a message to the appropriate process. A distributed implementation of each driver would route a message to the appropriate process on the appropriate processor. In both cases, this is hidden from the model writer who simply uses an *emit* or *consume* operation.

It should be noted that while providing the appropriate level of abstraction will allow a distributed simulation driver to be inserted with no changes to the interface for the model writer, it will not solve the basic technological problems of distributed simulation. These problems, such as that of clock synchronization, must be addressed with a layered or non-layered approach.

3.5. Battle Management Representation

The SADMT effort was driven by the need to evaluate a variety of battle management approaches in a standard simulation framework. Existing simulations did not adequately represent battle management algorithms usually providing a single "implied" approach to battle management embedded in the simulation of the architecture.

The SADMT Simulation Framework provides a capability of a hierarchical representation of the battle management processes. The process abstraction allows aggregates or *arrays* of processes to be defined. Techniques for assigning processes to execution environments and having the resulting performance reflected in the simulation are also provided. In addition, assertions about the interactions of processes can be specified and verified at runtime.

DETEC, EV-88, and EVP also provide a mechanism for the explicit specification of Battle Management algorithms. However, assignment to an execution environment and assertions are not provided.

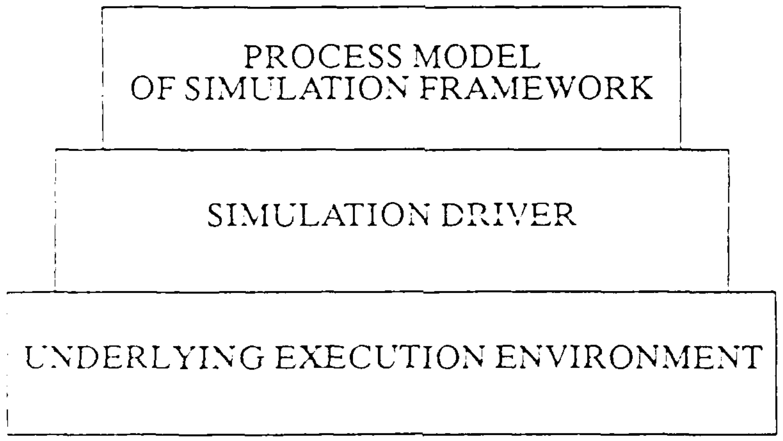


Figure 1. Layers of Simulation Support

4. DEVELOPMENT PLAN

In order to achieve the required functionality of the SDI Simulation Framework, the strengths of the existing efforts must be drawn upon to ensure the use of the best technology currently available. The development plan must address the establishment of an original standard framework, as well as a structure to support its evolution.

4.1. Initial Operating Capability

DETEC is the only simulation framework currently available that (1) supports a framework with visability of modeling assumptions, (2) currently has a reasonable size library of algorithms and technology models, and (3) provides a user interface to support the use of this library and evaluation of simulation results. It therefore provides the best baseline from which to evolve and should be the initial version of the SDI Simulation Framework.

4.2. Evolution Issues

It is important, however, that the SDI Simulation Framework continue to evolve, making use of new technological advances in software engineering, simulation, and distribution techniques. In the remainder of this section, the key issues in the evolution of DETEC are addressed.

4.2.1. Level of Abstraction for Simulation Operations

The level of abstraction of simulation operations on the SDS Simulation Framework must hide the implementation detail from a model writer. For instance, in SADMT a model writer can call a WAIT procedure. Behind the scenes, in the driver, the operation of placing an event on a queue to cause a wake-up at the appropriate time is done for him. These queueing operations are not seen or available to the model writer. To change the current central queue to a distributed one does not, therefore, change the interface to the model writer.

DETEC, however, currently provides a lower level of abstraction, requiring a model writer to call the routines to place an event on the event queue and schedule a wake-up. The queueing mechanism is visible to the model writer. Thus a change in the DETEC interface is required to move to a different (possibly distributed) implementation of the driver.

4.2.2. Communication Modeling

A very important aspect of a simulation framework is how well it manages the communications aspects of the architecture being specified/simulated. Specifically, how well does the specification technique capture what process interconnections are allowable at any instant in time and what mechanisms are present in the driver to help ensure that only allowed communications take place? DETEC provides appropriate facilities by which the user can specify enabling conditions for communications between platform-level processes; further, the DETEC driver monitors communication attempts for "physical viability".

SADMT is somewhat weaker here in that only the mechanisms for suppressing physically invalid communications are provided; it would be straightforward to implement DETEC's notion of viability as a default but this has not yet been done. SADMT does, however, provide additional mechanisms for (1) detailing internal-BMC^3 communications within a platform and (2) supporting assertions on the contents of messages that are visible to the driver.

EXEC communications function at a lower level– that is, communications in EXEC take no cognizance of the fact that the messages being passed are part of an SDI simulation. Thus, its capabilities might be most effectively utilized in a distributed message-passing layer below the level of the simulation driver. In fact, EXEC is actually a distributed communications framework rather than a simulation framework since the aspects of communications management that are specific to architecture simulation are not present in EXEC.

4.2.3. Distribution

Once the SDS Simulation Framework model has evolved, as discussed above, the implementation and insertion of a distributed driver can proceed in parallel with the continuing use of the Framework and the development of a library of models. A thorough evaluation of the alternative approaches and their support of the types of interaction expected among SDI processes must be performed to ensure the use of the most appropriate technology. The experience and knowledge gained in the development of EV-88, the EVP framework, and EXEC should be used.

4.2.4. Link to Design Tools

The link between the design process and the SDS Simulation Framework is key in developing the required productivity and traceability. The dataflow process model and Ada template approach of SADMT supports a straightforward mapping between existing design tools and the simulation notation. This approach must be used in the implementation of the new process model of the SDS Simulation Framework.

4.2.5. Language for Interface Specification

The use of Fortran 77 by DETEC provided it with two key advantages: execution speed and compiler stability. However, the advantages of the Ada language cannot be overlooked. The scoping technique of Ada allows (1) the interactions (and non-interactions) of models to be verified, and (2) the enforcement of restricted access to the internals of the simulation driver. These are key points in verifying a simulation. The strong typing of Ada provides an automated approach to checking interface consistency. This supports the evaluation of designs, identifying key inconsistencies early in the life-cycle. In addition, the modern programming language facilities available in Ada support the development of modeling code.

The design point for the SDS Simulation Framework that optimizes the advantages and disadvantages of the available programming languages is one in which:

(1) interfaces specifying the structure of the design are written in Ada (as in SADMT),

(2) the simulation driver is implemented in the most appropriate language, based on compiler availability, performance requirements, and life-cycle cost, and

(3) model semantics are implemented in the most appropriate language, based on compiler availability, performance requirements, the availability of existing code, and life-cycle cost.

5. CONCLUSIONS

The steps required to address the establishment and evolution of the SDS Simulation Framework are :

(1) establish DETEC as the Initial Operating Capability

(2) specify the new abstract process model implemented in Ada that (a) supports a high level of abstraction for simulation primitives, (b) provides a model of communication that supports distribution, and (c) supports a link to the design process.

(3) specify guidelines for writing technology models and algorithms that minimize any changes between the current DETEC model and the new SDS Simulation Framework model, and

(4) in parallel (a) develop a library of technology models and algorithms and (b) implement the distributed simulation driver for the SDS Simulation Framework.

To: J. Dominitz

From: D. W. Fife, IDA-CSED [illegible]

Date: 2 February 1988

Subject: Alternatives for SDL and Tools

This note outlines for discussion: 1) a position on a System Description Language, SDL, for SDI architecture representation to meet POET goals; 2) a set of minimum requirements for computer-aided design and documentation tools to support the SDL and assist POET in formulating and communicating its recommendations to SDIO, agencies/elements, and future contractors.

SDL Scope and Objectives

The SDL proposed here is a set of graphical/textual presentations that complement MIL-STD-490 Specification Practices and DOD-STD-2167A Software Development Standard. The objective of SDL is to serve as a long-term standard for communicating SDI architectural/design information to a wide audience of system and software engineers. Consistent with recommendations of the recent Defense Science Board Report on Military Software, it is tool and vendor independent and based on widespread and competitive commercial practice. This means that the SDL is NOT formulated to be a specialized design methodology and language that would exclude all other tools, languages, and design techniques.

SDL is intended to support system design review, validation, and acquisition management. The audience which will use SDL is NOT REQUIRED to be well-versed in a unique methodology/language such as IDEF, TAGS/IORL, or others. The SDL presentations rather are intended to be as commonplace as feasible, given the technical substance to be communicated. A straightforward one page description should suffice to explain each presentation form and its rules. Thus, SDL would be defined by a series of one page guides covering each form.

Minimum Required Presentation Forms

The following lists the graphical/textual forms of SDL. Any other self-evident diagrams, tables, and forms may complement these. Those below will be defined by one page guides as suggested.

** DATA/CONTROL FLOW DIAGRAM – an iconic diagram distinguishing external interfaces, data flows, control flows, databases, and data transforming processes.

** CONTROL/PROCESS ACTION TABLE – a decision table defining and relating input data conditions and the resulting output data or control actions of each control or data transforming process, including the required response time values.

** DATABASE STRUCTURE DIAGRAM – a graphic identifying database record types or data item group types and their defined interrelationships.

** DATA DICTIONARY LIST – a listing by database record type or data group of all the contained data items and their characteristics, including units of measure.

** CRITICAL SEQUENCE DIAGRAM – an iconic diagram showing an end-to-end series of processes, with timings, which comprises a critical path in system function.

** RESOURCE/FUNCTION ALLOCATION – an iconic diagram identifying the system's physical elements, by type and quantity, and showing their physical interconnections such as communications, the "logical" control or data transforming processes allocated to each, and information on resources available and consumable by the assigned processes.

Tool Requirements (Preliminary)

IDA's current tracking of computer-aided tools convinces us that a highly competitive market is producing major tool enhancements and, consequently, comparative cost-benefit realignments among tools. POET would NOT be well advised to recommend a specific tool other than for its direct use over the next six months. A better approach is to define minimum tool requirements to guide a tool selection by the contractor or staff that will directly use the tool for POET's support. These requirements also would help tool producers set their enhancement goals so as to assist SDI programs better in the future. The following requirement statements are intended to permit a competitive selection among a reasonably large set of COTS tools.

Supporting Hardware/OS Platform. The computer-aided design software (the "tool") should be hosted on a graphic workstation microcomputer, operating the Unix System V or Berkeley 4.3 operating system software and providing high resolution graphics on a minimum 15 inch diagonal screen. The Sun Microsystems 3/50 or equivalent represents the intended category of hardware.

User Interface. The tool should provide a three button mouse, iconic menu(s), and keyboard entry for the creation of diagrams representing system (hardware and software) designs. Displayable diagrams must include two-dimensional physical and spatial depictions, and the SDL diagrams. Multi-column, multi-row ruled and labeled tables must be displayable.

If any single diagram or table is not completely containable on the screen, the tool should provide smooth and easy scrolling to bring any portion to the center of the screen with one mouse pick action.

No specification element, graphic artifact, or relationship that may exist in the tool's design database (see below) shall ever be invisible on any type of display or diagram by which it may be created or modified, nor shall it ever appear visibly as identical to another distinct type of element, artifact, or relationship, nor shall it ever appear visibly to exist in the database when in fact it does not. The tool shall affix a type label to every specific display to indicate unambiguously the visible specifications that can appear on it.

Graphic Annotations. The tool should provide the facility for a user to define distinct, special icons with user-determined meaning, that can be placed in multiple instances anywhere on diagrams and tables comprising a design specification. The existence of these graphical notes shall be maintained in the tool's design database (see below), and shall be accessible on the same basis as any database element, for diagnosis, analysis, reporting, or printing.

Algorithmic Specification and Code Generation. The tool should support extended specification beyond SDL requirements by user text entry of a high level language such as SADMT, Ada, Common LISP, or FORTRAN. This capability shall be supported by template generation, see below, and may support automatic code generation as stated under Design Simulation. Extended text specifications should be associated by a tool with the SDL Data/Control Flow Diagrams and Action Tables.

Design Database. The tool should maintain all user-created design and specification information for a given project in an identified, integrated database. The vendor-determined structure and definition of the design database shall be published in user manuals to support user-prepared analysis and reporting software. The tool should include a library of database access programs to assist users in preparing their own analysis and reporting software.

The tool and its underlying host hardware/software shall permit the creation of a single system design as large as 1 Gigabyte of physical database space, provided the necessary disk equipment is attached to the system.

The tool should provide a database load/dump utility program to import or export a database in a published flat ascii file format. This supports portability and reuse on other hardware and database installations, which are not necessarily hardware and software compatible with the source tool. This utility also should load files conforming to the published format into the tool's database.

Text and Table Entry and Template Generation. The tool should provide for both standard (default) and user-definable edit template displays for all text entry exceeding one line of characters. The generation of templates should be based, whenever appropriate, on automated retrieval of pertinent design database information and its automatic inclusion in the template.

Database Inquiry and Reporting. The tool should provide a facility for user-defined queries of the design database.

The tool should be able to produce a printed listing of data dictionary items selected by various criteria, supporting the SDL requirements as a minimum.

Diagnostic Facility. The tool should provide for online and background (batch) analysis of the design database for a given system, for completeness and consistency. The user should be able to diagnose: a given diagram or table; a series of diagrams or tables that are related by being decomposed from a parent diagram or table; or, the entire design database. The tool should provide for the incorporation of user-defined diagnostic checks as part of the tool's diagnostic facility. In diagnostic checks on a given diagram, the tool should use all relevant database information to ensure that no errors or inconsistencies exist in data or control item references. Diagnostic messages should clearly and unambiguously indicate the specification element or artifact that causes the error.

Design Simulation Capability The tool should provide for automatically generating a simulation of a design to the level represented in the SDL Data/Control Flow Diagrams and Action Tables, as a minimum. This would be a flow-token simulation, but timing specifications must be recognized and the simulation output shall depict the number of flow actions, events, or firings occurring per unit of simulation time in every process or flow.

The tool also should provide functional simulation by automatically generating SADMT text (version 1.5) and/or DETEC text, to the extent that processes, data flows, and control/algorithmic specifications have been provided by the user/designer. This also provides a flow-token simulation capability for designs where full algorithmic specification has not been done.

Team Support and Version Identification. The tool should provide both user specified and automatic version identifiers and also date and time stamps for different versions of the design databases of a given project.

The tool should provide for design control in team activities by user access passwords and a locking/protection facility. The latter allows only designated users to modify a given diagram or table, but allows certain others, with authorized passwords, to view or read diagrams they are not allowed to modify.

Documentation Support. The tool should provide a facility to automatically produce input files to COTS publishing software from its design database and other text files on the host Unix software/hardware. This facility should recognize user-defined document templates as well as vendor-provided standard templates for documents meeting DOD-STD-2167A content/outline requirements. Diagrams and tables prepared via the tool, including SDL output forms, should be includable as distinct document elements.

Alternatives for POET Architecture Composition and Tools

No now-deliverable COTS tool meets all the requirements listed above as SDL and tool requirements, nor is any likely to evolve to meet this specific list within the next six months. POET must make a trade-off to set a mandatory requirements list and a desirables list, but can also "buy" (from the vendor or by a support contractor's effort) some special enhancements. Here are some initial examples of alternatives available.

Strong Design Validation Tool: Consider only tools with algorithmic specification already built in. These tools, such as TAGS, AUTO-G, and DCDS, can most readily meet the simulation requirements, including the SADMT or DETEC coupling. Because of their special design languages, they lack some SDL-like presentations, so "buy" some enhancements to produce those. DCDS, though undergoing a graphics upgrade for Sun, cannot meet the user interface requirements within POET's time schedule.

Lowest Cost Acceptable COTS Tool: Formulate the tool and SDL requirements as a minimum mandatory list, with additional desirables having predetermined points toward a selection score. Request vendors to bid. Reject those failing on any of the mandatory list, and select from the remaining tools one that has a 75 percent or greater score on desirables, at the lowest price. Pay for any additionally desired upgrades as need demands them.

Best Interim Choice Available: Set up a scored evaluation sheet incorporating all the above requirements, with some added details to help resolve scores assigned, and then assess tools as they are deliverable off-the-shelf on 1 March. Select one tool for POET's use only for an interim period, based on the highest score. Publish the complete requirements for interest and (hopefully) self-financed upgrades by tool vendors in the future. Provide advice to future contractors, e. g. SE&I, from the complete tool requirements specification.

Supported COTS Tool Evolution: Prepare a mandatory and desirables requirements specification, incorporating firm SDL and tool requirements, with a specified budget for enhancements to be made by the winning vendor while POET uses vendor's present, unenhanced tool on a temporary basis. Advertise and get bids, score them on scope and quality of the enhanced end product, and select the one giving the most cost-effective proposal.

September 15, 1988

Recommendations on Distributed Operating System R&D for the Strategic Defense System

Karen D. Gordon
Cathy Jo Linn

Computer and Software Engineering Division
Institute for Defense Analyses
Alexandria, Virginia

ABSTRACT

The Strategic Defense System (SDS) imposes a set of requirements on distributed operating systems that is not met by state-of-the-art systems. In this report, the key requirements are identified as being: (1) real-time support, (2) reliability/fault tolerance, and (3) security. The extent to which these requirements are being addressed by current distributed operating system research is discussed.

The major SDIO-funded distributed operating system projects – Alpha, Cronus, and Mach – are reviewed, compared, and evaluated. A fourth project, the V distributed system project of Stanford University, is also highlighted, because of its unique potential for meeting certain SDS needs. Recommendations on the directions in which the SDIO should pursue each of these projects are made.

The ONR Real-Time Systems Initiative, which is addressing some of the most pressing needs of the SDS, is described. It is recommended that the SDIO seek to coordinate with the ONR in this effort.

1. SDS DISTRIBUTED OPERATING SYSTEM REQUIREMENTS

The SDS imposes a set of requirements on distributed operating systems that is not met by state-of-the-art systems. The most critical, and at the same time highest risk, requirements can be captured under the following headings: (1) real-time support, (2) reliability/fault tolerance, and (3) security.

Before proceeding with a discussion of each of these classes of requirements, let us stress that the SDS is not envisioned as a monolithic system under the the control of a single, universal distributed operating system that must meet all of these stringent requirements at once. At the highest level, it is seen as at least two systems: (1) the SDS development and maintenance system, and (2) the deployed SDS system. In turn, the SDS development and maintenance system will consist of interconnected heterogeneous networks of heterogeneous systems. The individual systems will include both uniprocessor and multiprocessor configurations, some of which may run system-unique operating systems. The deployed SDS system will also consist of interconnected networks of uniprocessor/multiprocessor systems; however, the degree of heterogeneity will be controlled. These two systems – the SDS development and maintenance system and the deployed SDS system – place very different demands on distributed operating systems, in particular, on the real-time, reliability/fault tolerance, and security aspects of the systems.

Although it is not clear how far the boundaries of individual distributed operating systems can or should extend in the SDS, it can be presumed, given the state of the art, that the boundaries will extend far enough to encompass the computing resources of a local area network or of a space-based platform. Of course, distributed operating systems will incorporate capabilities for communication with each other, as well as with other SDS components.

1.1. Real-Time Support

Real-time response is the singlemost critical requirement of distributed operating systems for the deployed SDS. "Functional" or "logical" correctness of a result without timeliness of the result is useless; in fact, the system that produces the result is "incorrect" if it does not meet timing constraints.

Real-time requirements include the following:

- o Real-time support must extend to aperiodic tasks, as well as to periodic ones.
- o Real-time support must provide for stability under transient overload. That is, it must ensure that the least "important" tasks are the ones that miss their deadlines (or otherwise suffer) first.
- o Real-time support must extend to all system resources. In particular, it must extend beyond single processors, both to multiprocessing and distributed processing architectures, and to other shared logical and physical resources. Priority inversion, in which a lower priority task blocks a higher priority task, must be minimized and accounted for.
- o Real-time support must not ignore the fact that failures are bound to occur. It must enable system operation and performance to degrade gracefully as components fail.

Real-time response is not a mandatory requirement of the SDS development and maintenance system.

1.2. Reliability/Fault Tolerance

Reliability and fault tolerance are broad concepts. At the highest level, reliability/fault tolerance can be viewed as addressing two distinct but inter-related concerns: data integrity and processing integrity. Data integrity deals with ensuring that data can survive failures – that data is not lost, corrupted, or made inconsistent. Processing integrity deals with ensuring correct and continuous processing.

To consider the reliability/fault tolerance requirements of the deployed SDS, let us focus on two aspects: (1) ground-based command and control, and (2) space-based operations. The ground-based command and control functions place high demands on both data integrity and processing integrity. That is, correctness and continuity of both data and processing is vital. The space-based platforms, on the other hand, cannot afford the redundancy that is required to achieve absolute data and processing integrity. In particular, they have to be very judicious in applying physical redundancy, which can mean idle resources, and in applying temporal redundancy, which can mean time delays. Their goal is to "optimally" apply all available resources in performing their mission.

1.3. Security

Given the state of the art, it is not clear how far the concept of multilevel security can or should extend in the SDS. That is, while multilevel secure distributed operating systems are often postulated and would offer certain advantages, they are not mandatory.

The obvious alternative to multilevel secure operation is system-high operation. In system-high operation, all data or objects in the system are treated as if they were classified at the highest level allowed in the system. That is, only subjects cleared to the system-high security level can access *any* of the objects in the system. For example, in a Top Secret system-high facility, objects of Confidential, Secret, or Top Secret classification can be stored, but only Top-Secret-cleared subjects can access any of the data. (In a multilevel secure facility, Confidential subjects could access the Confidential objects; Secret subjects could access the Confidential and Secret objects; etc.) Authentication and access control can be implemented by physical measures (e.g., guarded vaults) or by automated measures (in accordance with the Orange Book).

System-high operation is an acceptable mode of operation for the deployed SDS. Moreover, it is believed that the implementation of multilevel security could impose insurmountable performance penalties on real-time response. Therefore, at least in the short term, SDS architectures that impose the requirement of real-time, multilevel secure distributed operating systems should be avoided.

While multilevel secure distributed operating systems may not have a role to play in the deployed SDS, with its emphasis on real-time response, they do appear to be more promising and more desirable in the SDS development and maintenance environment. In this environment, data of all security levels will exist, and the number of personnel involved will be large. Global system-high operation is undesirable and probably infeasible. Penalties would be incurred in one of two ways. Either there would have to be an inordinately large number of (high) personnel clearances; or some qualified personnel would end up being denied access, and their expertise would be lost.

An alternative to global system-high operation would be to have a separate system for each security level. For example, there could be a Secret system and a Top Secret system, each operated in system-high mode. This solution sacrifices unification of resources. For example, the same Secret data might appear in both systems. Maintaining consistency in such a case would be at best cumbersome. If the Secret data did not exist in both systems, then a Top Secret subject would be forced to access both systems, which is also undesirable.

Other alternatives for the SDS development and maintenance system, which focus on providing unification of specific resources include: (1) separate system-high systems, interconnected with multilevel secure communication networks, which provide for unification of communication resources; and (2) multilevel secure data management (for example, via the integrity lock approach, which utilizes cryptographic checksums), which provides for unification of data.

2. OVERVIEW OF CURRENT DISTRIBUTED OPERATING SYSTEM RESEARCH AND ITS APPLICABILITY TO THE SDS

Current distributed operating systems research is, for the most part, focused on the LAN-based interactive-user-oriented environment. Emphasis is on supporting the users – providing them a unified system with a convenient programming environment. In regard to SDS requirements, the following generalities summarize the extent to which the requirements are addressed by current research:

- o In an interactive-user-oriented environment, the workload is ad hoc; that is, the workload is whatever the users choose to offer. The resource management and control is, out of necessity, general-purpose. Typically, the goals are to minimize delay (e.g., response time) or to maximize throughput. In particular, real-time support is a non-issue.
- o Mandatory (Orange Book) security has also been viewed as a non-issue. Most prototype distributed operating systems exist in academic computing environments, which are not known for their security consciousness.
- o However, reliability and fault-tolerance have received much attention, at least in terms of data integrity. Mechanisms such as data replication, atomic transactions, and nested transactions continue to be explored in great depth.

This interactive environment presents challenges similar to those presented by the SDS development and maintenance environment. In particular, user support and data integrity are both key requirements in the SDS development and maintenance environment. Therefore, current distributed operating systems research is directly applicable to the SDS development and maintenance system. The major shortfall lies in the area of security.

However, the deployed SDS, especially the space-based platform environment, presents different challenges. Instead of being ad hoc, the workload (i.e., the software that is run) is ultimately well-defined. Convenience to users is of secondary concern. The primary challenge is to meet the real-time requirements of the application in a hostile environment.

3. REVIEW, COMPARISON, AND EVALUATION OF MAJOR DISTRIBUTED OPERATING SYSTEM PROJECTS

In this section, the major SDIO-funded distributed operating systems projects are reviewed, compared, and evaluated: (1) Alpha, which is being developed at Carnegie-Mellon University under RADC sponsorship, (2) Cronus, which is being developed at BBN, also under RADC sponsorship, and (3) Mach, which is being developed at Carnegie-Mellon University under DARPA sponsorship. In addition, the V distributed operating system project of Stanford University, which is sponsored in part by DARPA (but not with SDIO funding), is presented, because of its well-established position in the distributed operating systems field and its potential for meeting SDS requirements.

3.1. Alpha

Of all the distributed operating system efforts, Alpha is the one whose stated goals most closely match the requirements of SDS space-based platforms. In particular, Alpha's target domain is distributed, real-time BM/C^3 systems.

Alpha can be characterized by two key concepts: the object/thread programming model and time-driven resource management. Although time-driven resource management is often presented as being inseparable from the object/thread programming model, it is in fact separable. In particular, as discussed below, time-driven resource management is to be incorporated into Mach, which utilizes the process/message/port programming model.

Object/Thread Programming Model

The object/thread programming model of Alpha was adopted from the Clouds distributed operating system project of Georgia Tech. The Clouds project is still actively investigating the support and exploitation of this model. However, emphasis in the Clouds project is on reliability and fault tolerance (as opposed to real-time support).

Time-Driven Resource Management

The concept of time-driven resource management was formulated by Doug Jensen, Doug Locke, and Hide Tokuda in the context of CMU's Archons Project (which dates back to 1979 and of which Alpha is the operating system effort). In time-driven resource management, both the time constraints and the relative importance of each computation (i.e., thread) are specified, by a time-value function. Time-value functions, in conjunction with best-effort scheduling, are notable in that they offer a means to deal with (1) aperiodic tasks; (2) transient overload (which is bound to occur, under stress conditions, when it is actually most important for the system to perform its mission); and (3) soft deadlines.

The time-driven resource management concept is also being pursued in the context of Mach, by Hide Tokuda. The philosophy is different, however. In Alpha, time-driven resource management is applied to both periodic and aperiodic tasks. In Mach, processor time is dedicated to periodic tasks (effectively giving them priority over aperiodic tasks), and time-driven resource management is applied only to the scheduling of aperiodic tasks. In other words, the Alpha philosophy is to work in the aperiodic domain; periodicity is viewed as an artifact of (in Alpha's view) out-dated approaches to real-time system development. The Mach philosophy, on the other hand, is to proceed traditionally, from the periodic domain.

Time-driven resource management is computationally expensive, and its practicality and effectiveness remain unproven. The assignment of importance values to tasks, for example, has not been adequately addressed. Further research is needed to convincingly demonstrate the applicability of time-driven resource management to SDS problem domains.

As previously noted, time-value functions and best-effort scheduling offer means of dealing with aperiodic tasks and transient overloads, both of which must be handled in the deployed SDS. However, alternatives do exist, and are being pursued in earnest in the context of the ONR Real-Time Systems Initiative, which is discussed below. Most of the alternatives are based on fixed-priority, rate-monotonic scheduling. They incorporate extensions for both aperiodic tasks and transient overloads. Furthermore, they address the synchronization of shared resources and the resulting potential for priority inversion.

Status

Alpha is in an early stage of development. Work has concentrated on the kernel; system services have not been implemented, and programming support is minimal. In regard to the kernel, time-driven resource management has been limited to processor scheduling. Reliability/fault tolerance, which is viewed as being a vital part of the Alpha kernel, is still in the process of being designed and implemented, as part of ongoing Ph.D. thesis research at CMU.

Concerns

Time-driven resource management is an intuitively appealing approach to real-time resource management. However, as discussed above at length, its general viability and its applicability to the SDS are uncertain.

The goals of the Alpha project are indeed well articulated, and, furthermore, align with many of the goals of the SDS. However, in the view of the distributed operating system research community, the accomplishments of the project are more modest. Publications are limited; and, at the same time, much system development remains to be done, despite the number of years that have been spent in the problem domain.

3.2. Cronus

Status

The Cronus project has proved its concept, of integrating heterogeneous computer systems by imposing a layer of standardization (in the form of the Cronus distributed operating system) on top of (heterogeneous) native operating systems. The Cronus system is a mature system.

Alternatives for integrating heterogeneous computer systems exist, including: 1) Evolving ISO OSI standards, which are addressing distributed application development, and represent *international* standards for distributed application development; 2) Heterogeneous Computer System (HCS) project of the University of Washington, which is relying on emulation and accommodation of multiple standards, rather than resorting to new standards; and 3) Existing data communication protocols, which offer limited functionality (in the form of remote login, file transfer, and electronic mail), but which are (almost) universally implemented and may suffice in the short term.

Concerns

The Cronus system has not been applied in the general community (i.e., outside of RADC, BBN, and MITRE). Therefore, its effectiveness in supporting distributed application development remains to be convincingly demonstrated.

Cronus was designed to integrate heterogeneous *centralized* systems. Now, the challenge is to integrate heterogeneous systems, some of which are conventional centralized systems, but others of which are *distributed* systems (e.g.. Mach, V, Clouds, Alpha). It is not clear how well the Cronus approach extends to this type of environment.

In a Cronus-related RADC/BBN project, the Secure Distributed Operating System (SDOS) Project, the question of how to incorporate multilevel security into Cronus was investigated. The following conclusion was reached: "Thus, the host operating system(s) on top of which SDOS [i.e., *secure* Cronus] is implemented must have a minimum of a B2 rating, and ratings of B3 or A1 are more desirable." GEMSOS, a product of Gemini Computers, Inc., of Carmel, California, was recommended as *the* multilevel secure operating system upon which to base SDOS.

This recommendation (GEMSOS as *the* native operating system) seems antithetical to Cronus's chief purpose of integrating computer systems with *heterogeneous* operating systems. That is, to achieve multilevel security, a *homogeneous* base of multilevel secure native operating systems must be installed; heterogeneity, the raison d'etre of Cronus, is sacrificed in the pursuit of multilevel security.

3.3. Mach

Status

Mach has a solid technical foundation, due in part to its evolution from RIG and Accent, earlier projects of its principal investigator. Moreover, it has achieved a broad base of interest and support, due not only to its technical foundation, but also to its UNIX compatibility and strong backing from DARPA. Mach is serving as a platform for several interesting distributed system research efforts, many of which are aimed at the SDS requirements of real-time support, reliability/fault tolerance, and security.

Because of its solid technical base and broad base of interest and support, Mach represents a valuable resource to the SDS, especially as an operating system for the SDS development and maintenance environment, and possibly as an operating system for ground-based components of the deployed SDS.

Concerns

On the negative side, Mach remains too bound to Unix. Mach was produced by modifying and enhancing Unix, with ideas and experience gained from the RIG and Accent efforts. (Its critics claim that it simply brings Unix into the twentieth century, for example, through its advanced virtual memory concepts.) Although a kernelized version of Mach, in which Unix functionality and code (specifically, the file system) is removed from the kernel, has been planned for some time, it has not yet been implemented and delivered.

3.4. V

Status

The V project (led by David Cheriton) has a solid record of ideas and accomplishments, in the form of numerous publications and a mature system. The V kernel is the preeminent minimal kernel. Its only competitor is the kernel of the Amoeba distributed operating system, which is a joint effort between the Centre for Mathematics and Computer Science and the Vrije University, both located in Amsterdam, the Netherlands. (The Amoeba project leaders are Sape Mullender and Andrew Tanenbaum.)

The V kernel is widely recognized in the data communications community, as well as in the distributed operating system community, for its high performance, especially its high-performance interprocess communication.

The V kernel offers traditional real-time application support. That is, it incorporates features typical of current (centralized) real-time operating systems, such as strict priority-based scheduling, accurate time services, and memory-resident programs. In addition, it extends real-time support into the distributed system domain through interprocess communication features such as datagrams, prioritized message transmission and delivery, and conditional message delivery (in which the message is delivered only if the receiver is awaiting a message when the message arrives).

As a mature, high-performance, (traditional) real-time minimal kernel, the V kernel represents a promising base upon which to build a real-time distributed operating system suitable for the deployed SDS.

Concerns

The V distributed system is sometimes viewed as ignoring security and reliability/fault tolerance issues. This perception stems from the V emphasis on minimizing the kernel. However, minimizing the kernel is also one of the keys to achieving a trusted, secure system. Moreover, VMTP, the protocol underlying V interprocess communication, has facilities that can support both security and reliability/fault tolerance. These include "entity domains" and encryption, which can provide isolation between security levels, as well as multicast communication, which has proven to be helpful in implementing fault tolerance. The problem is that these approaches to security and reliability/fault tolerance have not been fully developed or implemented. This would involve work above the kernel; the underlying kernel mechanisms are already in place.

4. ONR REAL-TIME SYSTEMS INITIATIVE

Since real-time support is the most pressing SDS distributed operating system research issue, it is important to examine real-time support as an issue in itself. That is, it is necessary to look beyond mainstream distributed operating system research, and into the research of the real-time systems community. The point is to identify research relevant to the SDS, as well as to identify means of gaining leverage from it. It is in this vein that the ONR Real-Time Systems Initiative is addressed here.

In FY 89, the ONR is beginning a five-year Real-Time Systems Initiative. The objective is to establish a scientific foundation for distributed real-time system development, to remedy the current situation in which ad hoc practices prevail. The Initiative is capitalizing on previous ONR-sponsored work. Already, the Initiative is capturing the interests of the real-time systems community, and promises to be the focal point of real-time research and development over the next few years.

The Initiative is focusing on uniprocessor scheduling initially. Then, in later years, multiprocessor and distributed system real-time support will be addressed.

Key participants in the Initiative will include the Software Engineering Institute and Carnegie-Mellon University. Their work is based on fixed-priority, rate-monotonic scheduling, with extensions to address some of problems that are encountered in applying this type of scheduling in practice. Extensions include:

- Deferrable server algorithm, for dealing with aperiodic tasks. Basically, this algorithm preserves some processing bandwidth for aperiodic tasks, while ensuring that periodic tasks meet their deadlines.
- Period transformation method, for dealing with transient overload. The problem is that rate-monotonic scheduling gives the highest priority to the tasks with the shortest periods. In the case of overload, the lowest priority tasks, i.e., the tasks with the longest periods, will miss their deadlines first. But the tasks with the longest periods may actually be the most "important." The period transformation method allows long-period tasks to have artificially high priorities, by having them emulate multiple short-period tasks.
- Priority inheritance protocols, which deal with synchronization and attempt to avoid priority inversion. The basic idea is to have a task that is blocking other tasks execute at the highest priority of the blocked tasks.

5. RECOMMENDATIONS

Alpha

1. Alpha is most notable for its emphasis on the concept of time-driven resource management, which represents its most interesting aspect and potentially valuable contribution to SDS. However, Alpha is not alone in exploring the concept. Moreover, time-driven resource management is not as developed and mature a discipline as the traditional

periodic-based, priority-driven approach to real-time system development. At this point, both approaches should be pursued.

2. The Alpha researchers have assumed that the SDS problem domain is an ideal domain for the application of time-driven resource management. In order to assess the validity of this assumption, a prototypical SDS problem should be defined and cast in the time-driven resource management framework. The goals should include: (1) examining some of the fundamental assumptions of the Alpha philosophy, such as the assumptions that periodicity and priorities are "artifacts," (2) measuring the overhead incurred by time-driven resource management, and (3) demonstrating how importance values can be assigned to tasks in a methodical way.

Cronus

1. Cronus is ready to evolve out of the general research domain and into a specific problem domain. Further work should be undertaken in the context of a specific plan for utilizing Cronus in SDS work.
2. The National Test Bed (NTB) represents a potential domain for the application of Cronus. In particular, RADC and BBN have suggested that the appropriate role for Cronus to play in the SDS would be that of a distributed operating system for development and maintenance activities, such as those of the National Test Bed; and, according to RADC, Martin-Marietta has expressed interest in utilizing Cronus in the NTB. Therefore, the possibility of utilizing Cronus in the NTB should be pursued. Moreover, further work on Cronus should be undertaken at the direction of the NTB organizations, to ensure that the work is indeed useful to the NTB and, in turn, to the development of the SDS.

Mach

1. The kernelized version of Mach should be completed.
2. Since the payoff of having a multilevel secure distributed operating system, especially for the SDS development and maintenance system, would be high, research on incorporating multilevel security into Mach should be pursued.
3. Since there will be heterogeneous systems in the SDS development and maintenance environment, an approach for integrating Mach into a heterogeneous environment should be developed.

V

1. As a mature, high-performance, (traditional) real-time minimal kernel, the V kernel represents a promising base upon which to build a real-time distributed operating system suitable for the deployed SDS. The V kernel itself, as well as the experience of its developers, should be taken advantage of, to the extent possible, in the development of the SDS.
2. In particular, the V kernel should be considered as a distributed operating system base upon which to explore priority-based real-time scheduling policies, such as those being pursued in the ONR Real-Time Systems Initiative.
3. The possibility of encouraging the full development of V approaches to security and reliability/fault tolerance should be explored.

ONR Real-Time Systems Initiative

1. The ONR Real-Time Systems Initiative should be taken advantage of. At the least, its results should be followed and utilized as appropriate.
2. The possibility of utilizing SDIO funding to accelerate some of the ONR research, especially in the directions of multiprocessors and distributed systems, should be

investigated.

3. The possibility of defining prototypical SDS real-time system problems and offering them as applications to be addressed by ONR research should be investigated.

September 15, 1988

TO: Charlie Johnson

FROM: Cy Ardoin / IDA CSED

Re: Comments on Processor Panel Meetings of POET

This report provides an assessment of several of the projects presented to the POET processor panel.

NSA Secure BM Processing
TASK B41701

Security is a major issue in the deployment and development of an SDS system. However, the full effort of the NSA is impractical for the Phase-1 system. NSA would like all the system to operate at level B2 or better (Orange book); however, the turn-around time required to certify a system as B2 or better is impractical. Furthermore, current architectures suffer performance penalties from the access check required by such a system. Until architecture are designed with this stringent security in mind, running real-time system on such systems is impractical.

Ground-based system seem most apt to benefit from the work of NSA. Nevertheless the effort of NSA should focus on speeding up the certification process and assuring that certified systems can meet real-time requirements.

TASK B41701004 (Secure RH32) is a good idea. It attempts to place security into the design of the RH32. This could remove the performance penalty associated with access checks.

TASK B41701001 (Examine the Security features/potential of GP7) is a good idea. This must be known before such a system can be deployed in a ground based mode.

TASK B41701002 (Make the GP7 a Secure System): The funding level of TASK B41701002 cannot be determined until the previous task is complete. I have no reason to believe their figures. Furthermore, how will this architecture differ from USASDC's fault-tolerant Encore (they are both shared memory designs)?

TASK B41701003 (Acquire 2 GP7 Systems) is already started. I would have waited until the preliminary results of TASK B41701001 were available.

DARPA Parallel Algorithm/Demonstration
TASK B41804

SWAT: The algorithm examples are good; however, the study of alternate architectures does not have a near or medium-term payoff. Special purpose hardware is not practical for space based systems in the short-term.

CERPASS: Center for Parallel Architecture Research and Algorithms at NASA Ames. This may be redundant, the Algorithms group will have a better knowledge of the current algorithms work funded by the SDIO.

DARPA Parallel Architectures
TASK B41805

A useless presentation. I've had better architecture courses as an undergraduate. The brief was a high-level overview of all the parallel systems supported by DARPA. None of these systems support fault-tolerant or real-time systems. And none of them could be deployed in space during the near or medium-term. DARPA should fund fault-tolerant, real-time designs for the SDIO with emphasis on size, power, and radiation hardening.

RADC Processing Evaluations
TASK B41305

A system to simulate and evaluated the reliability of BM/C3 architecture configurations. I believe the system is to be delivered in Sep. 1988. Continued funding of this project should hinge on the evaluation of architectures proposed by the SSTS, BSTS, SBI contractors, and other real systems under development.

NASA Langly VHSIC Multiprocessing Tech
TASK B41501

GMOS is of dubious value. This is an approach at using Macro (I arge Grain) dataflow model of computation. The firing rules for this system are very restricted, it only allows "and" operations on incoming arcs and output on all output arcs concurrently. Basically, its a fork-join system. This fork-join system is not rich enough to support the scheduling requirement of an Ada run-time environment (despite the use of Ada in specifying the semantics of the individual nodes)

Further development of this model is of little value to the Phase-1 goals unless Phase-1 intends on using this restricted form of dataflow for all software in the SDS. This model of programming will require extensive retraining of programmers.

The designers claim that systems designed for GMOS have predicatable performance. This is true only in that the critical path of the fork-join network can be determined statically.

NASA Langly Advanced Information Processing System and
A Plan for Advanced Computer Tech for BM/C3
TASK B41501 and USASDC DASG 60-88-C0045

This is type of work is needed by the SDI. This was the only work of its kind presented to the Technical Panels. I think that more work of this kind is needed. In the medium and long-term, the SDIO should support the development of flexible processing architectures which satisfy the DoD needs. Currently, we have a limited variety of chips which satisfy DoD needs but each contractor must configure these chips to create a processing architecture for his particular needs. Funding the development of these processing architectures should reduce the overhead and risk associated with contracts and increase the reliability of space-based systems.

SDC Fault Tolerant Computing
TASK B41202

A fault-tolerant Encore may be useful for ground based systems. Are real-time issues addresses? If not, they should be addressed in the short term. Will the results be much different form NSA's work with the GP7?.

GENERAL COMMENTS

Radiation Hardened Memory should be a primary concern to the DoD and particularly the SDIO. Without such memories, the processing power of space-based systems is severly impaired. The weight and power requirement of low-volume (64K) chips is unacceptable. 256K and 512K chips are needed soon.

The lack of fault-tolerant, real-time parallel processors is the next major concern. DARPA has funded ”pedal-to-the-metal” fast machines; but, these machines fail to address the DoD concerns.

September 15, 1988

TO: D. J. Waddell

FROM: Cy Ardoin / IDA CSED

Re: Comment on the SDI Supported Work at Yale - Linda and Crystal

This report provides an assessment of the Linda and Crystal projects at Yale. This assessment outlines the benefits that the Linda and Crystal projects offer the SDI. The SDI program requires tools and methods for parallel processing. Linda and Crystal provide some of the capabilities needed in this domain. In this, they have direct engineering relevance to the SDI.

First, we consider the Linda project. This project is the one with the best probability of influencing the development of a SDS in the near or medium term. Because of this, continued funding is important. Linda offers several advantages:

1) Linda offers a quick and easy method for transporting existing Fortran and C codes to parallel machines.
2) Linda offers a unified concept of parallelism through Tuple Space; this simplifies the problems involved in producing parallel codes.
3) Linda is currently available on several different architectures and several different languages.
4) Methods for optimizing Linda operations exist and are currently being used.
5) Linda operations are relatively simple and can be customized to individual architectures.

Nevertheless, it is important that Linda take on more of an SDI flavor.

1) Some consideration should be given to the use of DOD languages. The implementor of Linda don't think a binding to Ada will be difficult; however, a binding will require close cooperation with a compiler vendor because of the overload resolution involved.
2) Linda should begin to consider BM/C3 problems that are important to the SDI instead of "commercial problems." The implementors are interested in developing Linda not in developing algorithms that utilize Linda.
3) Some consideration must be given to real-time, i.e., some way of ensuring a bound on communications delay and of utilizing some concept like priority must be incorporated into the model. The implementors are aware of this and are performing static flow analyses on the code in order to optimize communications. Certainly, there maybe operations (arbitrary pattern matches) which cannot be bound; but, these operation will have to be avoided by users.

The second project is the Crystal compiler and its associated run-time system. The Crystal project is a very important one. This type of of automated parallel code generator has several advantages.

1) The programmer can concentrate his efforts on the development of a high-quality parallel algorithm without worrying about a particular method of mapping his algorithm to a given architecture.
2) The paradigm used in the Crystal project for transformation via index set manipulation may be such an important one that many researchers in parallel code generation will want to use the same paradigm and share in the compilation framework. This would be a highly

desirable result since it may dramatically reduce the complexity of optimizing compilers while enhancing their portability. Customization of such systems is also simplified because of the regular nature of these transformations.

3) Architectural research can benefit greatly by this research. A retargetable compiler that produces high quality code for a variety of architecture can be used to assess architectural designs and changes to architectural designs.

4) The run-time techniques for load-balancing based on input-data dependencies are important in "pedal-to-the-metal" computing. Indeed, such methods sometimes allow the use of automatic techniques where this would otherwise be impossible.

Nevertheless, there are several areas that need attention within Crystal.

1) Crystal should be developed and adopted to some language as soon as possible. This will enhance its exposure and provide a better means for evaluating the system.

2) Some consideration should be given to the use of DOD languages. Once again, a binding will require close cooperation with a compiler vendor or some form of preprocessing.

3) Some consideration must be given to real-time. Currently, the researchers at Yale are investigating methods for bounding the performance of operation performed by the run-time. Some of these methods look promising, and other require still more investigation.

In summary, we recommend continued funding for both the Crystal and Linda projects. The future work for both project should stress the need to introduce them into the parallel-processing community as soon as possible. Both project are of value to the SDIO; techniques for developing and programming parallel algorithms are desperately need. Nevertheless, the SDI should also emphasize it own need (after all SDI is funding these projects); therefore, we should also stress the use of DOD languages and real-time requirements.

Distribution List for IDA Memorandum Report M-553

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
Lt Col Chuck Lillie SDIO Room 1E149 The Pentagon Washington, DC 20301-7100	2
Lt Col Jon Rindt SDIO Room 1E149 The Pentagon Washington, DC 20301-7100	2
Other	
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
CSED Review Panel	
Dr. Dan Alpert, Director Center for Advanced Study University of Illinois 912 W. Illinois Street Urbana, Illinois 61801	1
Dr. Barry W. Boehm TRW Defense Systems Group MS R2-1094 One Space Park Redondo Beach, CA 90278	1
Dr. Ruth Davis The Pymatuning Group, Inc. 2000 N. 15th Street, Suite 707 Arlington, VA 22201	1
Dr. John M. Palms, Vice President Academic Affairs & Professor of Physics Emory University Atlanta, GA 30322	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Keith Uncapher	1
University of Southern California	
Olin Hall	
330A University Park	
Los Angeles, CA 90089-1454	
Dr. C.E. Hutchinson, Dean	1
Thayer School of Engineering	
Dartmouth College	
Hanover, NH 03755	
Mr. A.J. Jordano	1
Manager, Systems & Software	
Engineering Headquarters	
Federal Systems Division	
6600 Rockledge Dr.	
Bethesda, MD 20817	
Mr. Robert K. Lehto	1
Mainstay	
302 Mill St.	
Occoquan, VA 22125	
Mr. Oliver Selfridge	1
45 Percy Road	
Lexington, MA 02173	
IDA	
General W.Y. Smith, HQ	1
Mr. Philip Major, HQ	1
Dr. Robert E. Roberts, HQ	1
Dr. John F. Kramer, CSED	1
Dr. Robert I. Winner, CSED	1
Ms. Anne Douville, CSED	1
Mr. Terry Mayfield, CSED	1
Dr. Cathy J. Linn, CSED	1
Dr. Cy Ardoin, CSED	1
Dr. Dennis Fife, CSED	1
Dr. Karen Gordon, CSED	1
Dr. Ostap S. Kosovych, STD	1
Mr. Albert J. Perrella, Jr., STD	1
Ms. Sylvia W. Reynolds, CSED	2
IDA Control & Distribution Vault	3